



LabVIEW Code Security

NI Test System Security Summit

Steve Summers

Security Lead

Aerospace & Defense Business Unit

NI

Meeting NIST 800-53 Controls

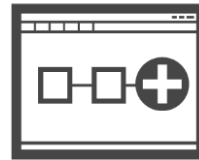
System owner must demonstrate that:

The organization requires the developer of the information **system**, system **component**, or information system **service** to...<meet the controls>

*On a deployed
LabVIEW System,
this means:*



Information storage



User-developed code

1011010
0101101
1011010

LabVIEW Run-Time Engine
Drivers



Hardware

This presentation

Meeting NIST 800-53 Controls

“The information system enforces policy where a subject that has been granted access to information can pass the information to any other subjects or objects; grant its privileges to other subjects; change security attributes on subjects, objects, the information system, or the information system's components; choose the security attributes to be associated with newly created or revised objects; or change the rules governing access control.”

“The information system does not release information outside of the established system boundary unless the receiving system or system component provides defined security safeguards; and security safeguards are used to validate the appropriateness of the information designated for release.”

“The organization requires the developer of the information system, system component, or information system service to identify early in the system development life cycle, the functions, ports, protocols, and services intended for organizational use.”

“The organization requires the developer of the information system, system component, or information system service to employ static code analysis tools to identify common flaws and document the results of the analysis.”

LabVIEW Code Security

Agenda:

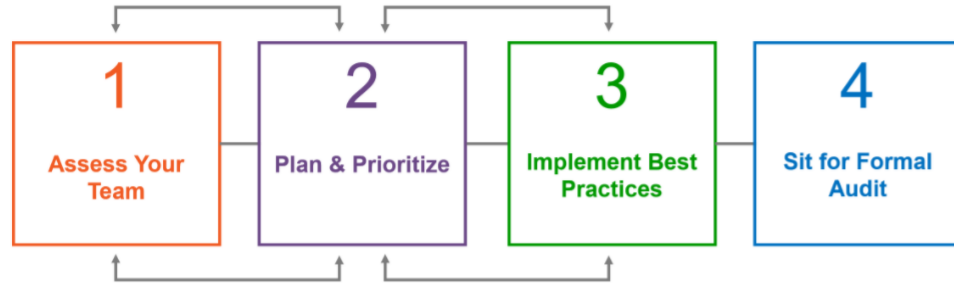
1. Develop quality code
2. Protect code development
3. Verify code performance
4. Validate code usage

Develop Quality Code



LabVIEW Center of Excellence

ni.com/lvcoe



Software standardization and success...

... entails more than ensuring a team uses the same software tool. Your team should be implementing consistent and proven software practices. Three pillars of expertise are essential for ongoing success: Engineer Good Software, Engage in Community Learning, and Ensure Technical Leadership.

These practices include ensuring that your architecture aligns to your application, that you have onboarding and appropriate training for your team, and that you implement software engineering processes that fit your workflow. Those teams that incorporate these best practices and choose to sit for and subsequently pass the CoE audit are awarded Center of Excellence Certification. Twelve teams world-wide enjoy this elite honor.

ASSESS YOUR TEAM

PROGRAM OVERVIEW

ENSURE TECHNICAL LEADERSHIP

ENGAGE IN COMMUNITY LEARNING

ENGINEER GOOD SOFTWARE

LabVIEW Style Guide

Included in LabVIEW Help for each version
 Tips to develop readable code
 Checklists to ensure compliance

Develop your own style guide

Style – Object-oriented, Actor Framework?
 SubVI terminal wiring standards
 SubVI documentation and icons
 Documentation guidelines

The screenshot shows the LabVIEW Help window with the 'LabVIEW Style Checklist' page open. The window title is 'LabVIEW Help'. The navigation bar includes 'Hide', 'Locate', 'Back', 'Forward', and 'Options'. The left pane shows a 'Contents' list with 'LabVIEW Style Checklist' selected under 'Application Development'. The main content area has the following structure:

LabVIEW Style Checklist

Use the following checklists to help you maintain consistent style and quality. You can customize these checklists to fit the specifications of your application.

Note The LabVIEW VI Analyzer Toolkit provides tests that check VIs interactively or programmatically for style, efficiency, and other aspects of LabVIEW programming. Refer to the [National Instruments website](#) to learn more about this LabVIEW toolkit.

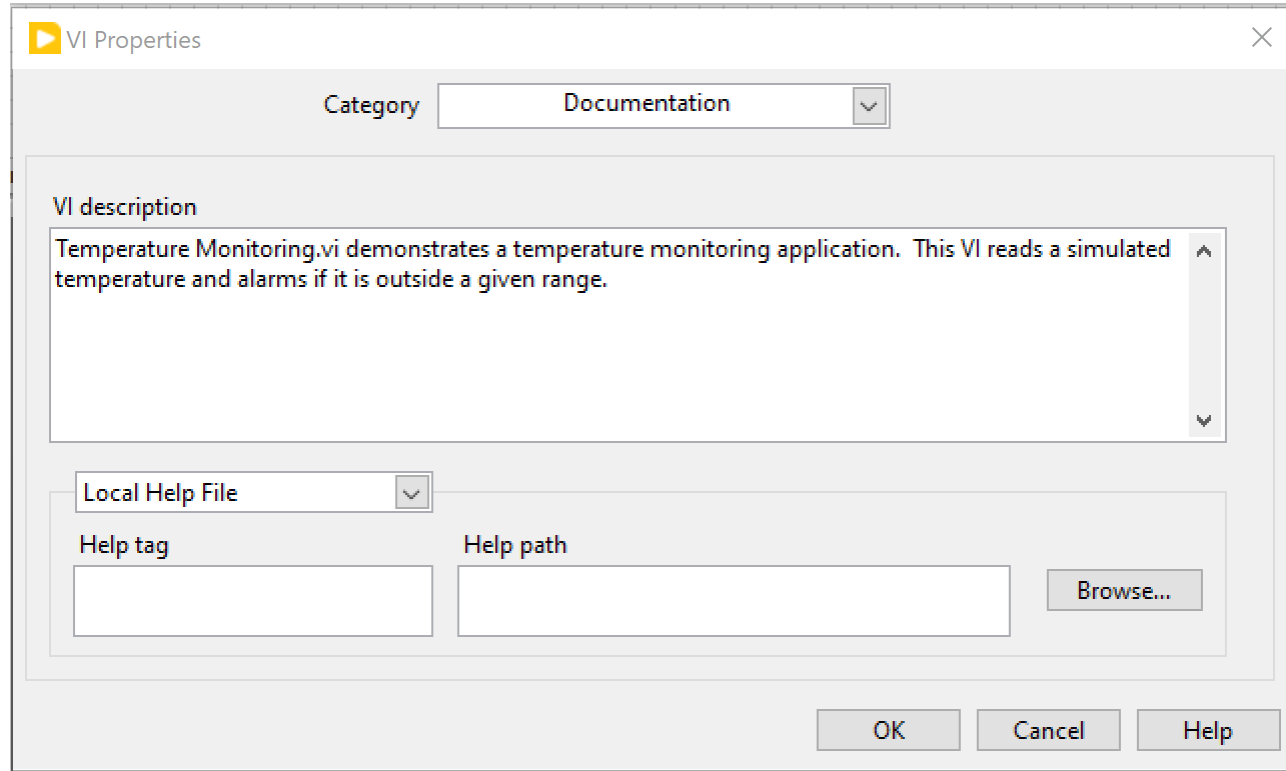
[Expand All] [Collapse All]

- Front Panel Checklist**
 - ▶ Show/Hide Checklist
- User Interface Front Panel Checklist** Top▶
 - ▶ Show/Hide Checklist
- Front Panel Dialog Checklist** Top▶
 - ▶ Show/Hide Checklist
- Conditional Checklist** Top▶
 - ▶ Show/Hide Checklist
- Non-User Interface Front Panels** Top▶
 - ▶ Show/Hide Checklist

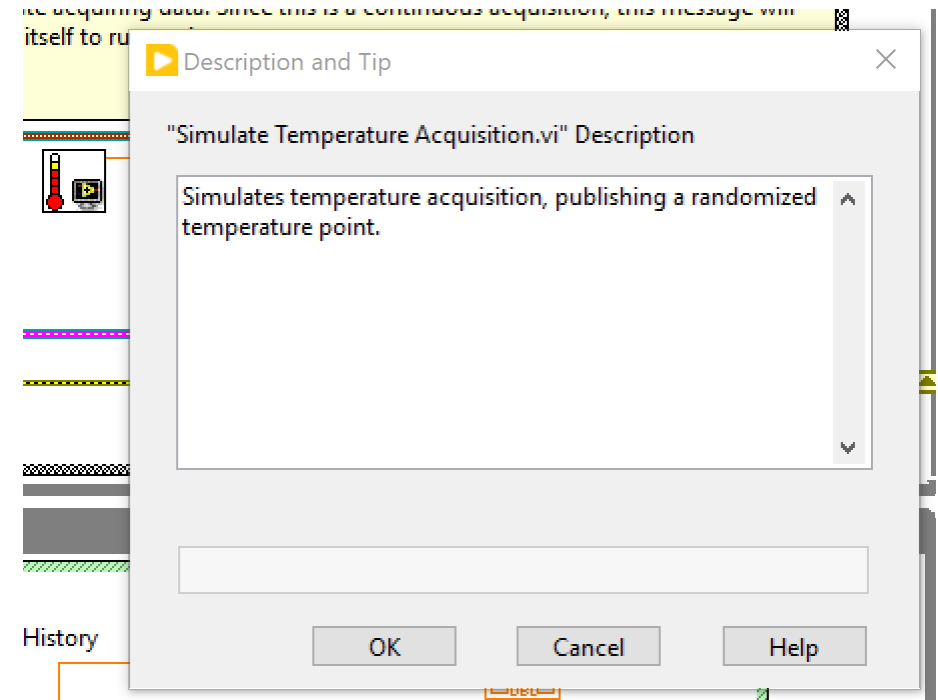
In This Topic

- ✔ [Front Panel Checklist](#)
- ✔ [User Interface Front Panel Checklist](#)
- ✔ [Front Panel Dialog Checklist](#)
- ✔ [Conditional Checklist](#)
- ✔ [Non-User Interface Front Panels](#)
- ✔ [Block Diagram Checklist](#)
- ✔ [LabVIEW Classes](#)
- ✔ [Labels and Comments](#)
- ✔ [Size and Location](#)
- ✔ [Programming Checklist](#)
- ✔ [Corner Cases Checklist](#)
- ✔ [Error Handling Checklist](#)
- ✔ [Performance Checklist](#)
- ✔ [Conditional Checklist](#)
- ✔ [SubVIs Checklist](#)
- ✔ [API Design Checklist](#)
- ✔ [Organization Checklist](#)
- ✔ [Project Organization Checklist](#)
- ✔ [Distributing Source Code Checklist](#)
- ✔ [RT-Specific Checklist](#)
- ✔ [Handling Data Checklist-RT](#)
- ✔ [Initialization and Shutdown Checklist-RT](#)

Document LabVIEW

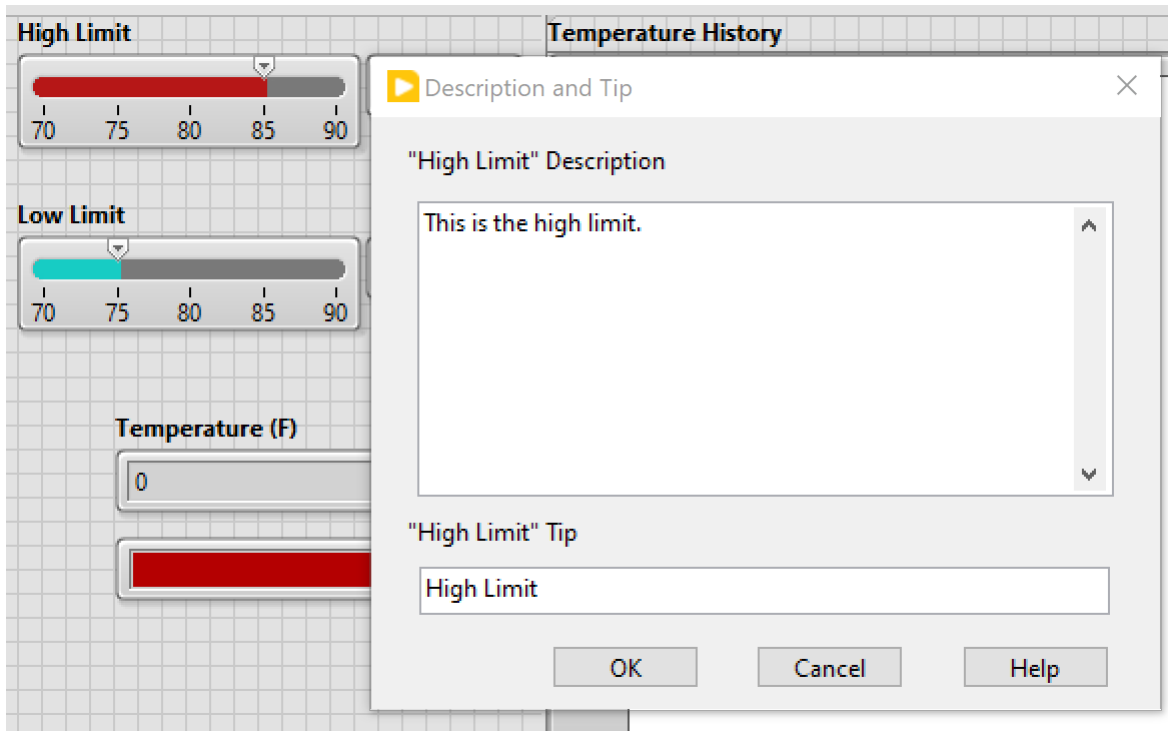


VI Properties...Documentation
 Overall VI Description
 Appears in all uses of the VI



Right-Click...Description and Tip...
 Appears only in that usage of the VI

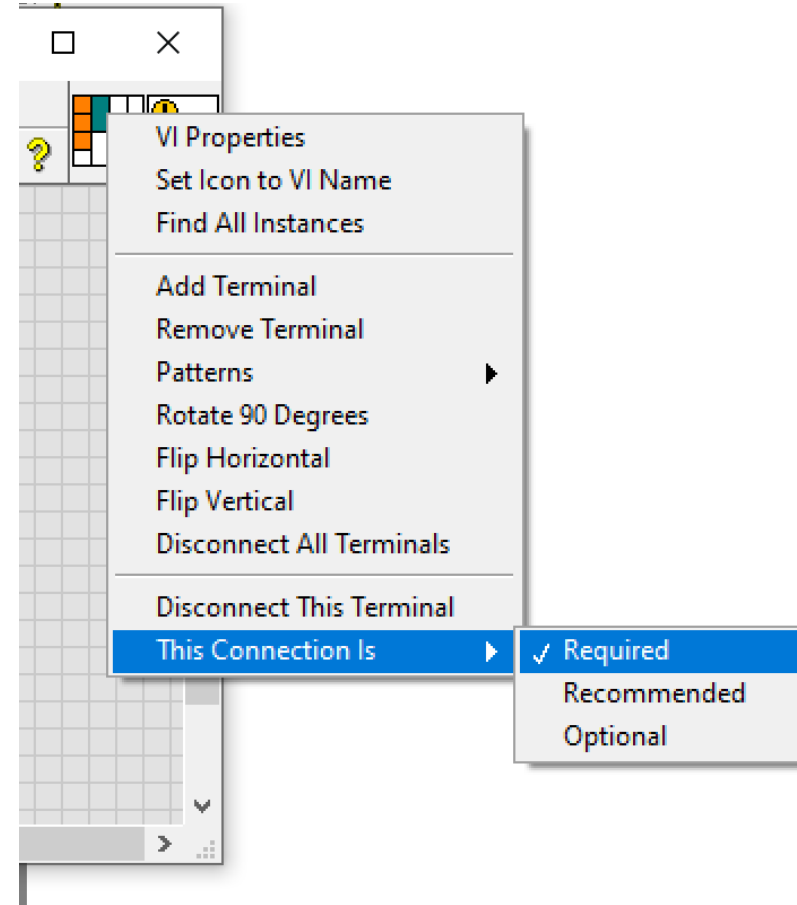
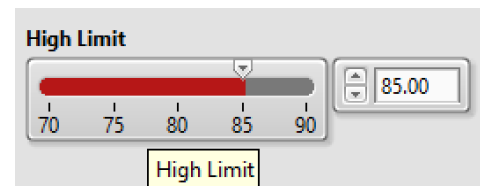
Document LabVIEW



Control...Right Click...Description and Tip

Description appears in VI Documentation

Tip appears in mouse-over



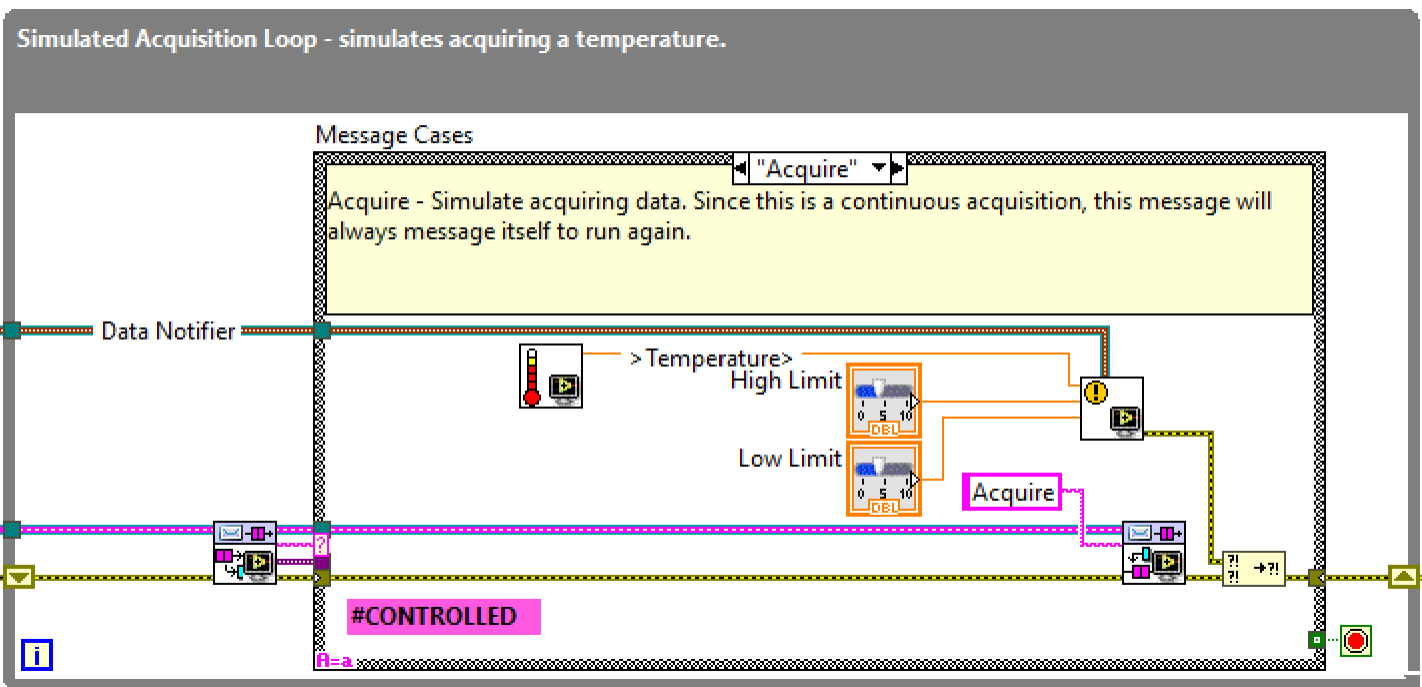
Terminals...This Connection Is...

Required

Recommended

Optional (Consider impact of using defaults)

Document LabVIEW



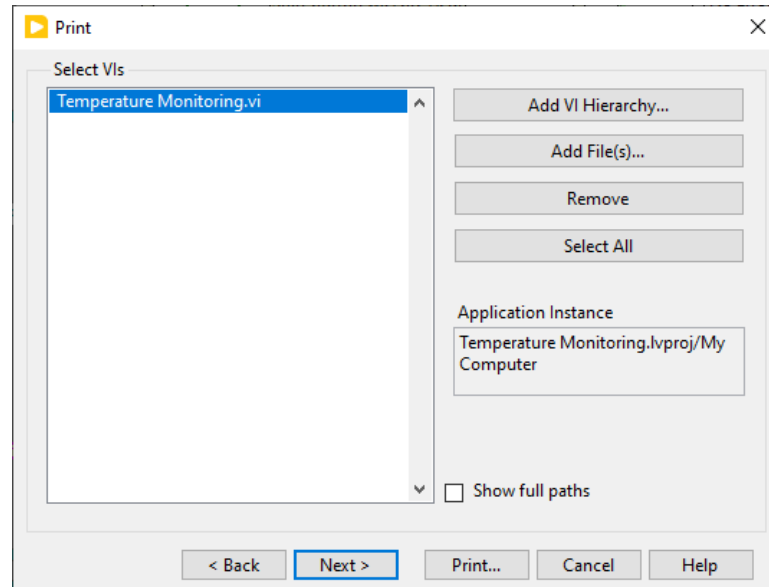
Visible Items...Labels

VIs, Functions, Wires, Loops

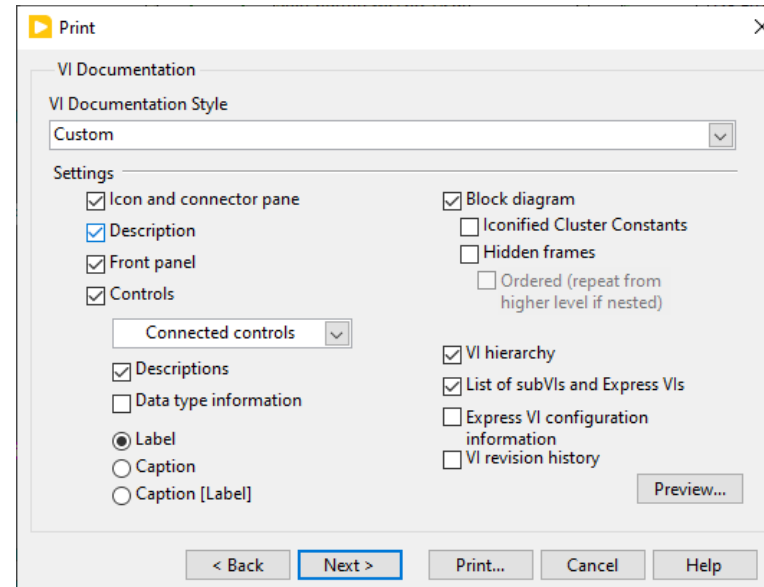
Use special labels to identify restricted code

Generate LabVIEW Documentation

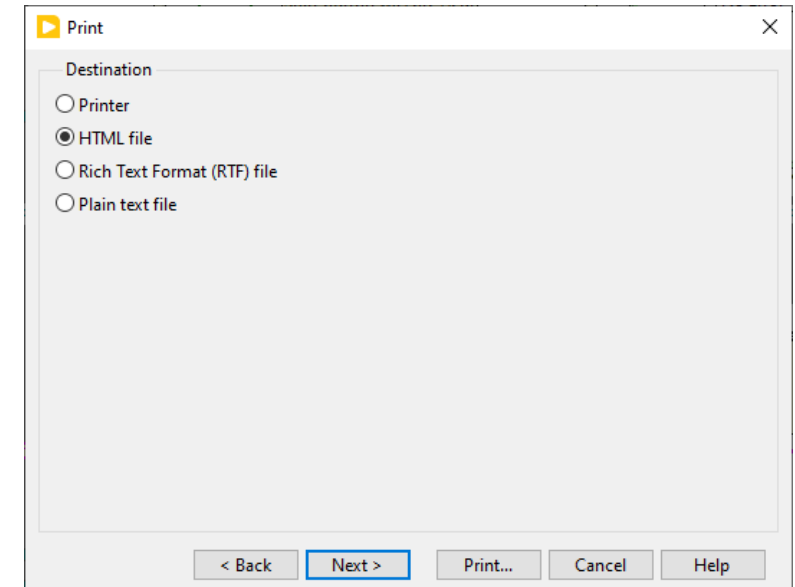
File...Print...VI Documentation...



Select Add VI Hierarchy for complete code

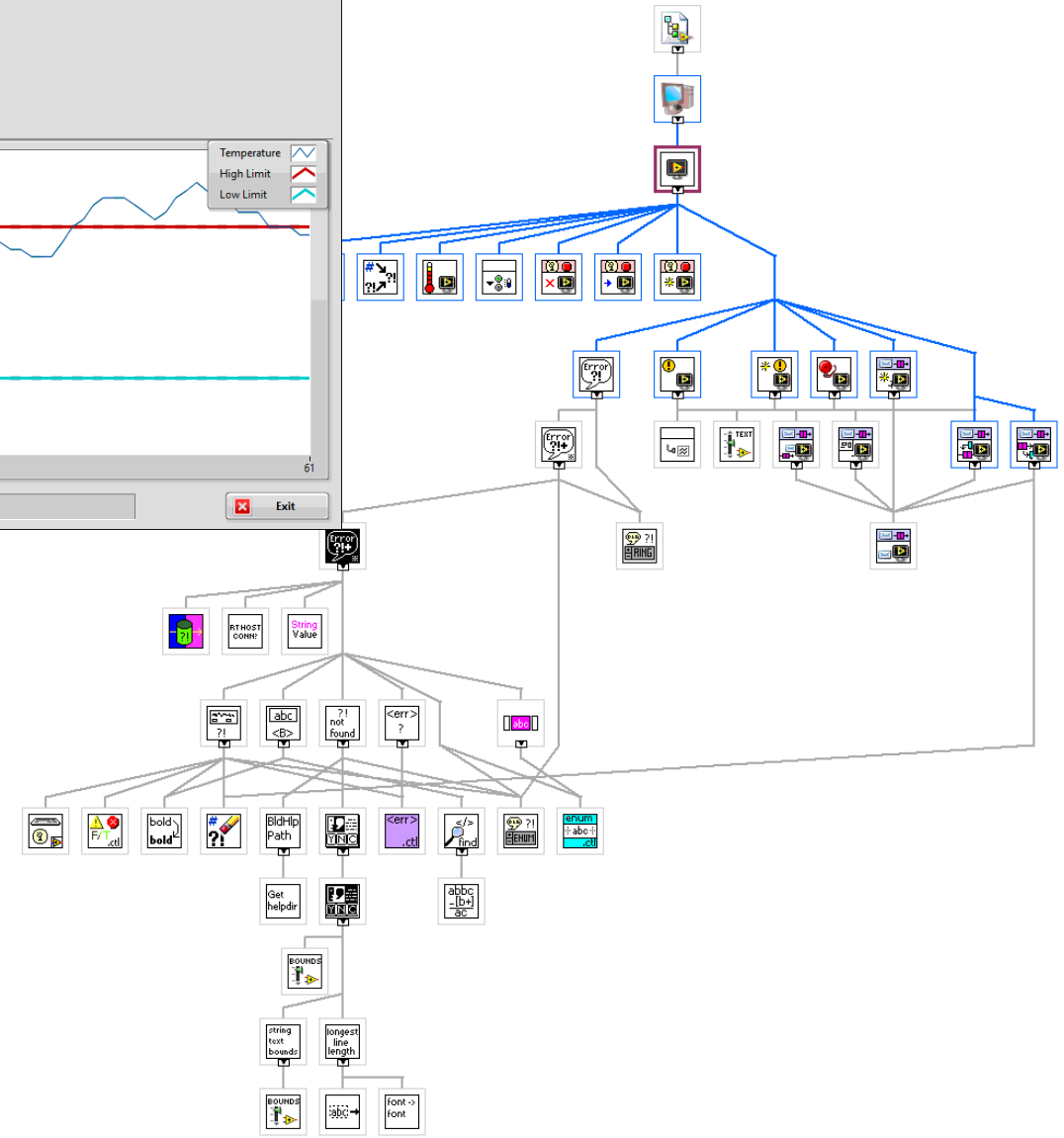
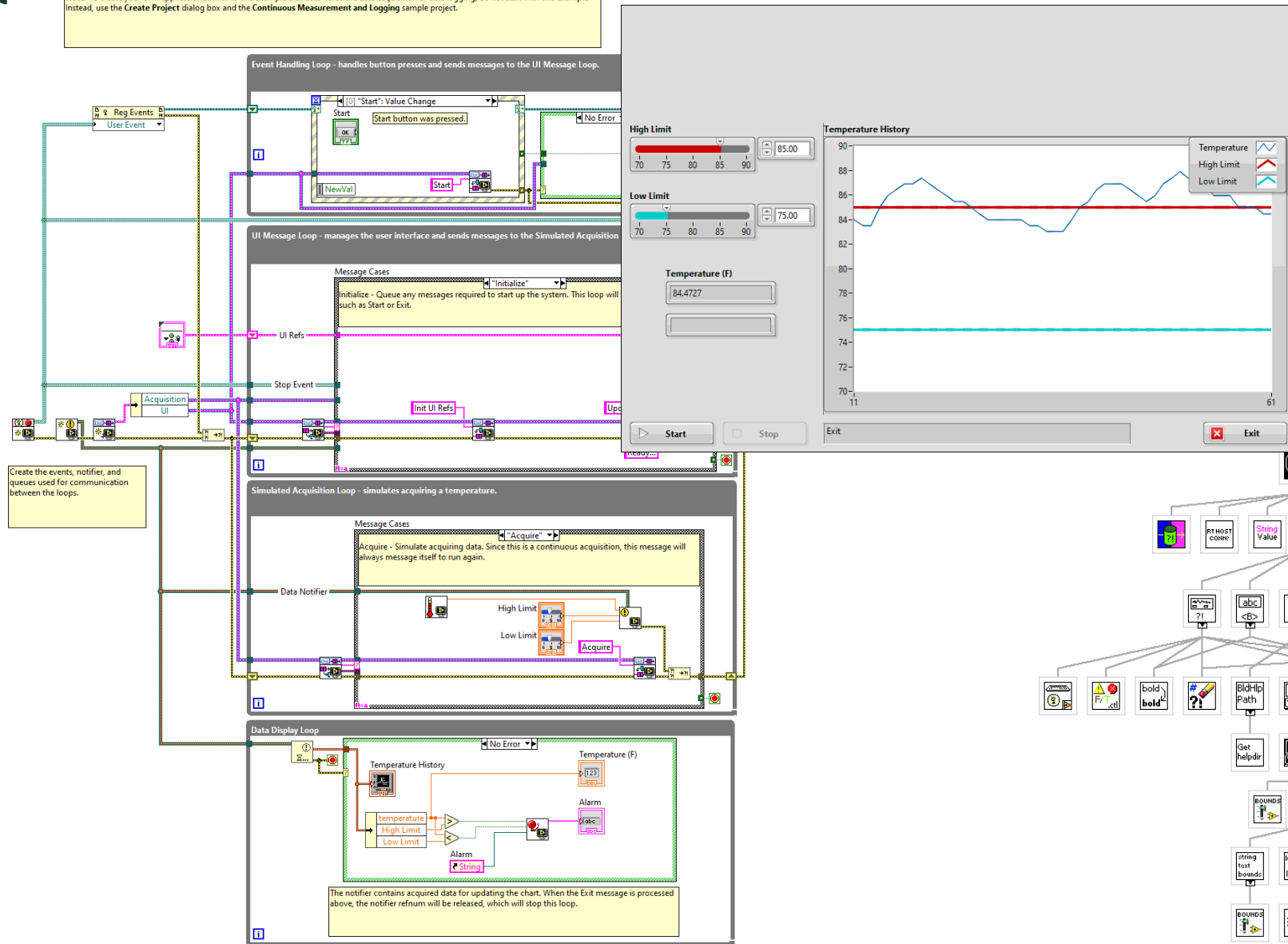


Set options for detailed documentation



Printer – Print to .pdf
RTF – Generate single editable document
HTML – Generates folder with images

Note: To create your own application similar to this example that uses hardware data acquisition or data logging, do not start with this example. Instead, use the Create Project dialog box and the Continuous Measurement and Logging sample project.

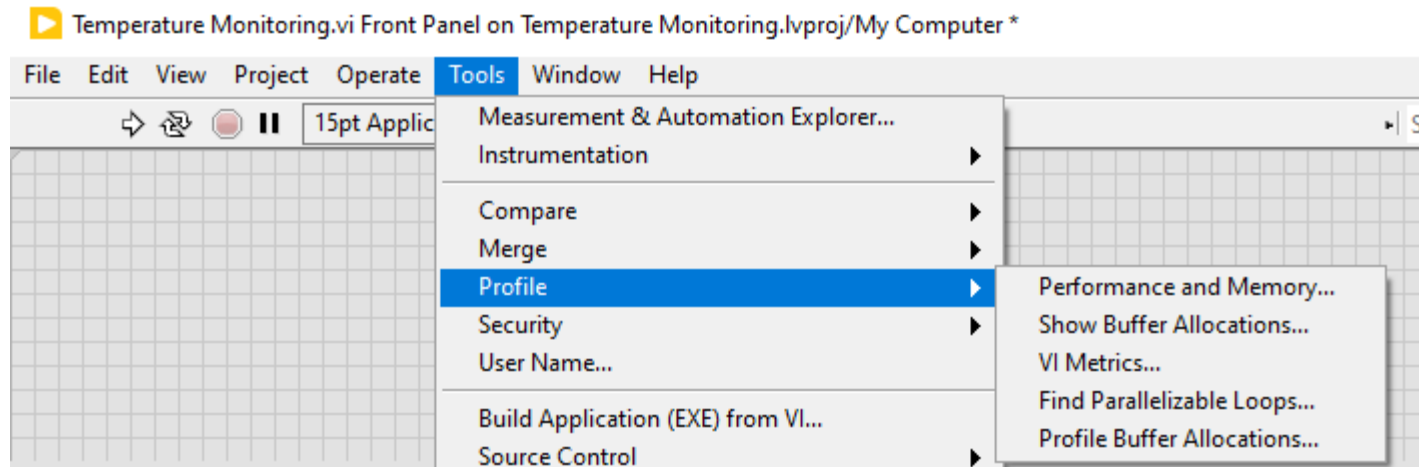


Block diagrams, Front Panel of each VI and SubVI

All comments

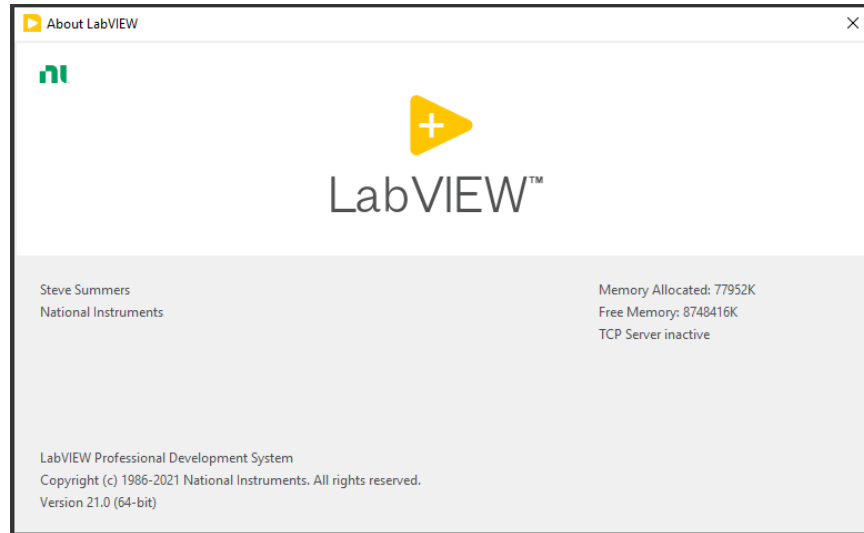
VI Hierarchy

LabVIEW Profile Tools



Free, included with all versions of LabVIEW

Memory Usage



Help...About LabVIEW

Total allocated memory in use

VI Name	VI Time	Sub VIs Time	Total Time	# Runs	Average	Shortest	Longest	Avg Bytes	Min Bytes	Max Bytes	Avg Blocks	Min Blocks	Max Blocks
Temperature Monitoring.vi	706.8	27.0	733.9	1	706.8	706.8	706.8	89.10k	89.10k	89.10k	67	67	67
Set Alarm Colors and Get Alarm Text.vi	24.8	0.0	24.8	30	0.8	0.1	2.3	5.15k	5.15k	5.16k	8	8	9
Temp Monitor Set Enable State on Multiple Controls.vi	0.7	0.0	0.7	6	0.1	0.1	0.2	3.76k	3.76k	3.76k	0	0	0
Send Data Notification.vi	0.5	0.0	0.5	31	0.0	0.0	0.0	4.54k	4.54k	4.54k	0	0	0
Temp Monitor Message Queue.lvlib:Dequeue Message.vi	0.4	0.0	0.4	41	0.0	0.0	0.1	3.23k	3.22k	3.31k	6	6	7
Temp Monitor Message Queue.lvlib:Enqueue Message.vi	0.3	0.0	0.3	42	0.0	0.0	0.0	5.50k	5.48k	5.59k	7	7	9
Simulate Temperature Acquisition.vi	0.3	0.0	0.3	31	0.0	0.0	0.0	4.51k	4.51k	4.51k	1	1	1
Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	0.0	0.0	0.0	2	0.0	0.0	0.0	3.69k	3.69k	3.69k	2	2	2
Temp Monitor User Event - Stop.lvlib:Create User Event - Stop.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	2.58k	2.58k	2.58k	0	0	0
Create Notifier for Simulated Data.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	3.73k	3.73k	3.73k	0	0	0
Temp Monitor User Event - Stop.lvlib:Fire User Event - Stop.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	2.57k	2.57k	2.57k	0	0	0
General Error Handler Core CORE.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	39.71k	39.71k	39.71k	49	49	49
Temp Monitor User Event - Stop.lvlib:Destroy User Event - Stop.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	3.67k	3.67k	3.67k	0	0	0
Temp Monitor Message Queue.lvlib:Create All Message Queues.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	5.72k	5.72k	5.72k	2	2	2
General Error Handler.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	13.48k	13.48k	13.48k	4	4	4
Simple Error Handler.vi	0.0	0.0	0.0	1	0.0	0.0	0.0	11.06k	11.06k	11.06k	5	5	5
Error Cluster From Error Code.vi	0.0	0.0	0.0	0	0.0	0.0	0.0	0.00k	0.00k	0.00k	0	0	0
Check Special Tags.vi	0.0	0.0	0.0	0	0.0	0.0	0.0	0.00k	0.00k	0.00k	0	0	0
Set String Value.vi	0.0	0.0	0.0	0	0.0	0.0	0.0	0.00k	0.00k	0.00k	0	0	0
Error Code Database.vi	0.0	0.0	0.0	0	0.0	0.0	0.0	0.00k	0.00k	0.00k	0	0	0

Tools...Profile...Performance and Memory

Memory, time per VI and SubVI

VI Metrics

VI	# of nodes	structures	diagrams	max diag depth	diag width (pixels)	diag height (pixels)	wire sources
-total-	235	20	56	-	-	-	304
-average-	18.08	1.54	4.31	0.92	583.46	385.85	23.38
Temperature Monitoring.vi	124	9	24	2	1189	1379	174
Create Notifier for Simulated Data.vi	5	0	1	0	232	288	4
Send Data Notification.vi	13	0	1	0	463	300	12
Set Alarm Colors and Get Alarm Text.vi	14	2	5	2	480	381	15
Simulate Temperature Acquisition.vi	10	1	2	1	643	300	10
Temp Monitor Message Queue.lvlib:Create All Message Queues.vi	11	1	3	1	725	325	15
Temp Monitor Message Queue.lvlib:Dequeue Message.vi	17	3	7	2	850	530	27
Temp Monitor Message Queue.lvlib:Enqueue Message.vi	12	1	3	1	740	476	16
Temp Monitor Set Enable State on Multiple Controls.vi	7	1	2	1	352	250	7
Temp Monitor User Event - Stop.lvlib:Create User Event - Stop.vi	6	1	3	1	618	260	8
Temp Monitor User Event - Stop.lvlib:Destroy User Event - Stop.vi	5	0	1	0	332	159	4
Temp Monitor User Event - Stop.lvlib:Fire User Event - Stop.vi	6	1	3	1	621	210	8
Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	5	0	1	0	340	158	4

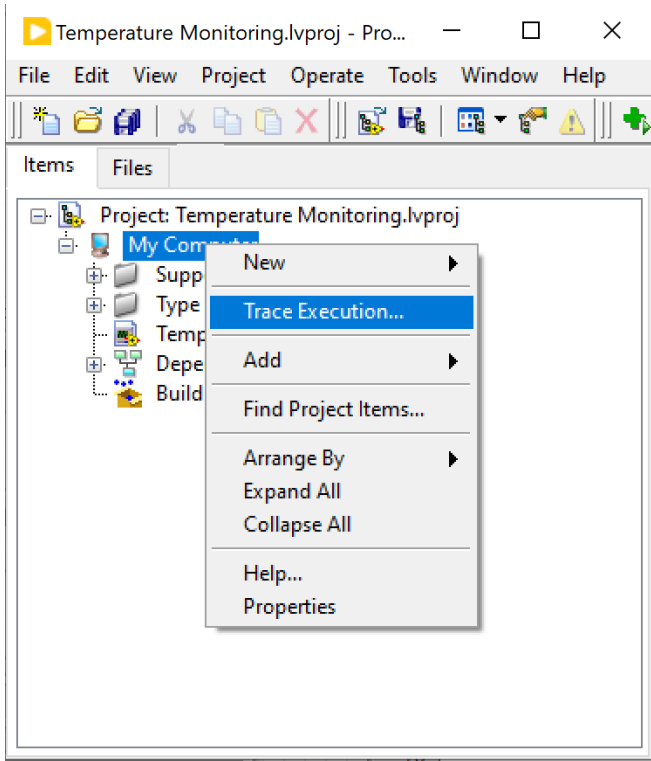
Tools...Profile...VI Metrics

Track code complexity

Track code changes over time

Desktop Execution Trace Toolkit

Included with Professional; separate install



Trace Execution...

Get details of program execution

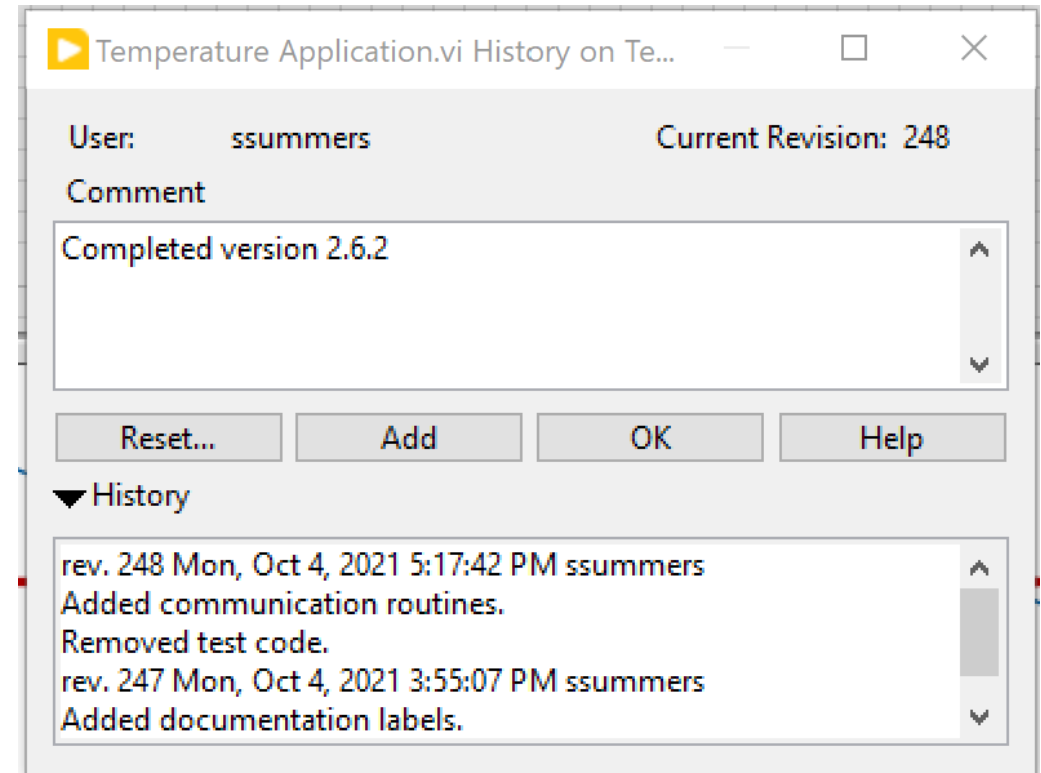
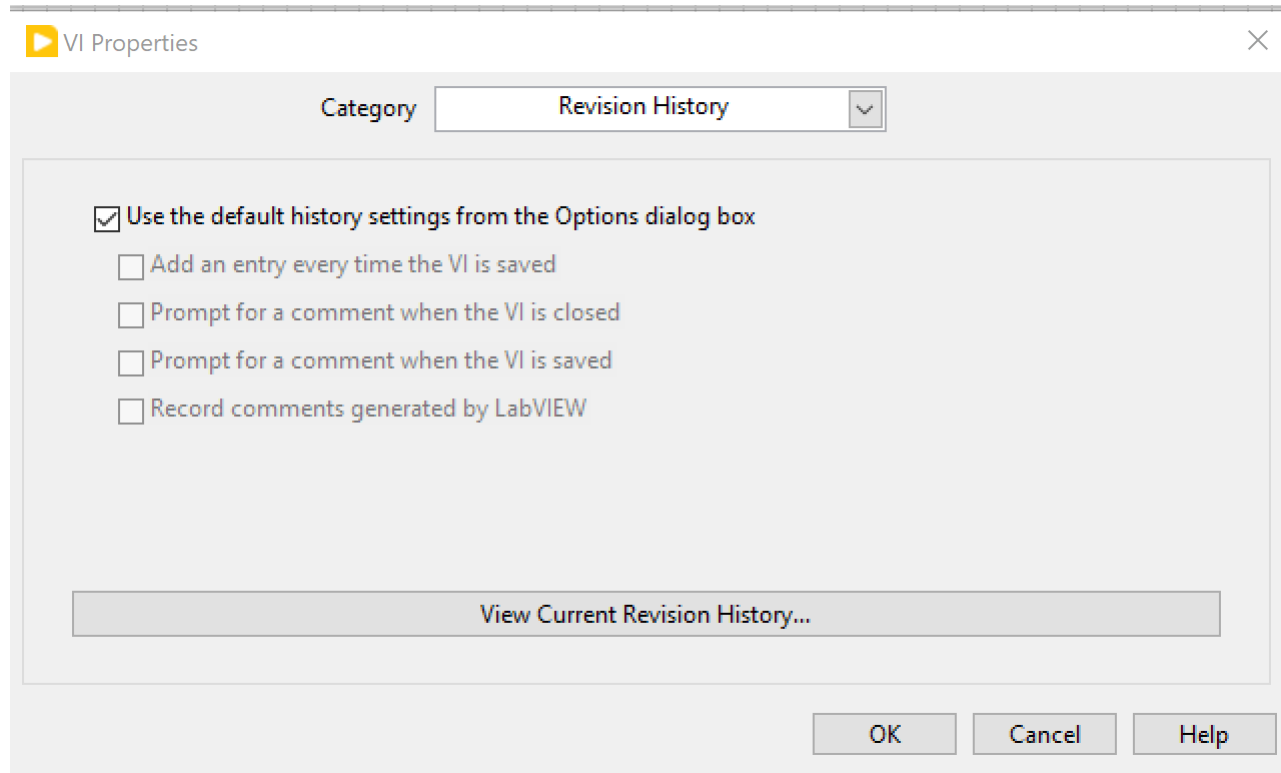
Look for unapproved actions

The screenshot shows the 'Desktop Execution Trace Toolkit' interface. The 'Target Configuration' section displays 'Machine: USAUSLT-XM8B04H' and 'Port: 3363'. The 'Application Instance' is set to 'Temperature Monitoring.lvproj'. The 'Trace Operation' section includes buttons for 'Capture Settings', 'Start', 'Stop', 'Clear', and 'New Trace'. Below this is a table of execution events.

#	Time	VI	Event	Thread ID	CPU ID	High
0	17:14:09.1001420	Temperature Application.vi	VI Start Execution	5	4	VI
1	17:14:09.1001501	Temperature Application.vi	VI Call	5	4	VI
2	17:14:09.1001572	Temp Monitor User Event - Stop.lvlib:Create User Event - Stop.vi:1600001	VI Call	5	4	Objec
3	17:14:09.1001634	Temp Monitor User Event - Stop.lvlib:Create User Event - Stop.vi:1600001	Create User Event	5	4	Objec
4	17:14:09.1001701	Temp Monitor User Event - Stop.lvlib:Create User Event - Stop.vi:1600001	VI Return	5	4	Objec
5	17:14:09.1001949	Temperature Application.vi	Register User Event	5	4	Objec
6	17:14:09.1001977	Create Notifier for Simulated Data.vi	VI Call	5	4	Objec
7	17:14:09.1002130	Create Notifier for Simulated Data.vi	Obtain Notifier	5	4	Objec
8	17:14:09.1002149	Create Notifier for Simulated Data.vi	VI Return	5	4	Objec
9	17:14:09.1002216	Temp Monitor Message Queue.lvlib:Create All Message Queues.vi	VI Call	5	4	Objec
10	17:14:09.1002244	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	VI Call	5	4	Objec
11	17:14:09.1002340	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	Obtain Queue	5	4	Objec
12	17:14:09.1002354	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	VI Return	5	4	Objec
13	17:14:09.1002364	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	VI Call	5	4	Objec
14	17:14:09.1002435	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	Obtain Queue	5	4	Objec
15	17:14:09.1002473	Temp Monitor Message Queue.lvlib:Obtain Message Queue.vi	VI Return	5	4	Objec
16	17:14:09.1002516	Temp Monitor Message Queue.lvlib:Enqueue Message.vi	VI Call	5	4	Objec
17	17:14:09.1002564	Temperature Application.vi	Wait on Notification	7	5	Objec
18	17:14:09.1002745	Temp Monitor Message Queue.lvlib:Enqueue Message.vi	Enqueue Element	5	4	Objec

Protect Code Development

Revision History



VI Properties...Revision History

Configure VI to require revision comments

View revision history

Compare LabVIEW Versions

Tools...Compare...Compare VIs

The screenshot displays two side-by-side LabVIEW windows. The left window, titled 'Temperature Monitoring.vi Front Panel', shows a control panel with 'High Limit' (85.00) and 'Low Limit' (75.00) sliders, a 'Temperature (F)' display showing '0', and 'Start' and 'Stop' buttons. The right window, titled 'Temperature Application.vi Front Panel', shows the same control panel but with the 'Temperature (F)' display showing '85.4492' and a red 'OVER TEMP' indicator. A 'Differences' dialog box is open in the center, listing 46 differences. The 'Details' pane shows the following information for the selected difference:

- Label Name: -----
- Object Type: Loop Tunnel
- Difference Type: Added
- Detailed Description: Added at (-77,1126)

The 'Include' section of the dialog box has the following checked options:

- VI Attributes
- Front Panel
- Position/Size
- Block Diagram
- Cosmetic

Compare LabVIEW Versions

Tools...Compare...Compare VI Hierarchies

Compare VI Hierarchies

First VI Hierarchy:
 Temperature Monitoring.lvproj\My Computer\Temperature Monitoring.vi
 [Browse Files...] [Browse Memory...]

Second VI Hierarchy:
 Temperature Monitoring.lvproj\My Computer\Temperature Application.vi
 [Browse Files...] [Browse Memory...]

Compare

- VI attributes
- Front panel
- Position/size changes
- Block diagram
- Cosmetic changes

[Read Hierarchies] [Compare Hierarchies] [Create Report] [Show Differences]

VIs in Hierarchies

- ✓ Set Alarm Colors and Get Alarm Text.vi
- ✓ Create Notifier for Simulated Data.vi
- ✗ Temp Monitor User Event - Stop.lvlib:Create User
- ✓ Temp Monitor UI Data.ctl
- ✓ Simulated Data.ctl
- ✗ Temp Monitor Message Queue.lvlib:Message Clu
- ✗ Temp Monitor Message Queue.lvlib:Obtain Mess
- ✗ Temp Monitor Message Queue.lvlib:All Message
- Simple Error Handler.vi

Description

VI Attribute - Miscellaneous

 Difference Type: Owning library

✗ different
 ✓ same
 ⊕ added to first hierarchy
 ⊖ deleted from first hierarchy
 ⊙ not compared
 ● shared VI

[Done] [Help]

Verify Code Performance

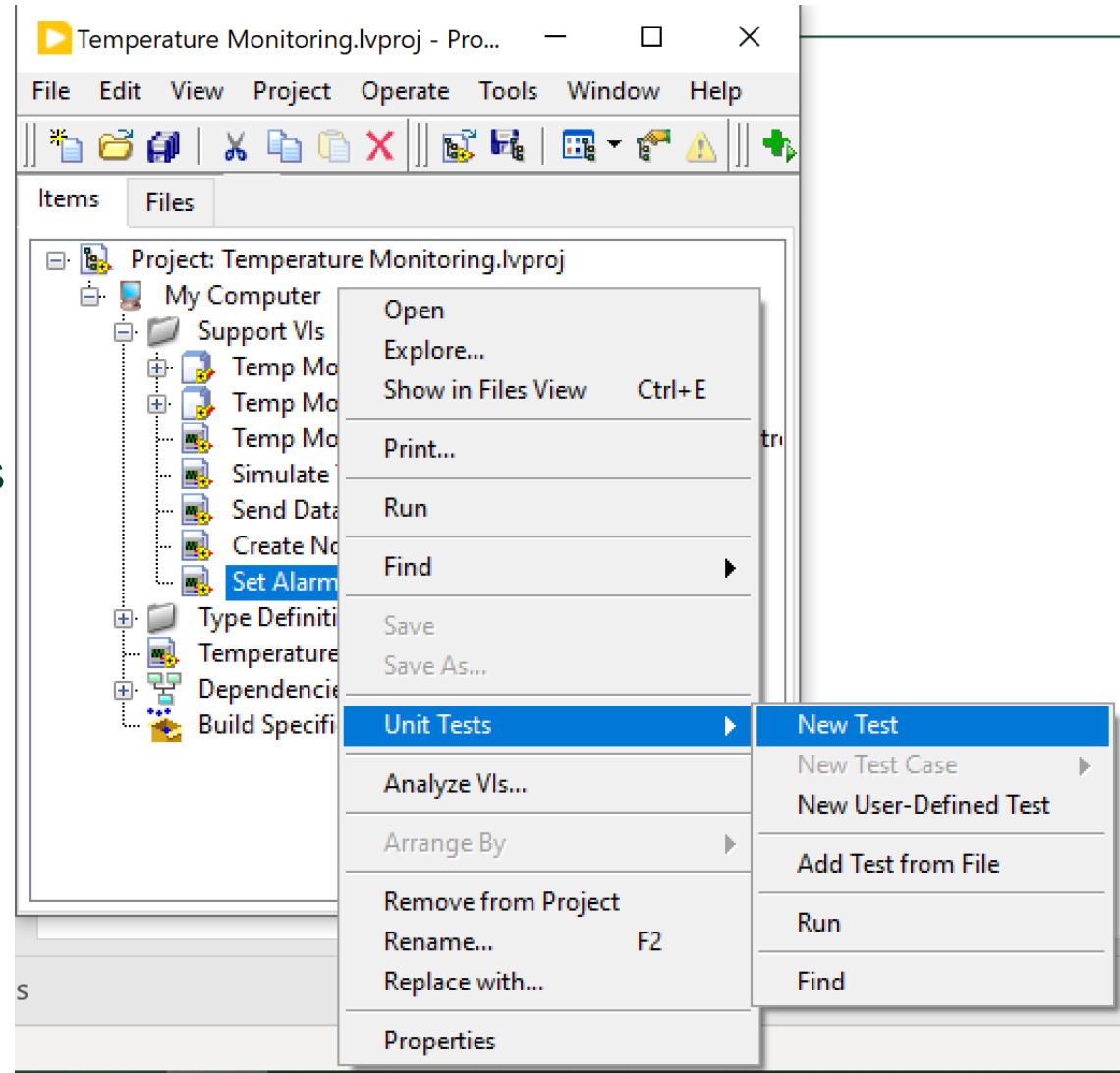
Unit Test Framework

Included with Professional; separate install

Project Window...Code...Unit Tests

Define Inputs, Expected Outputs for various test cases

Cascade to test code modules and hierarchies



- Category
- Configuration
- Test Cases**
- Setup/Tearardown
- Test Vectors
- Advanced

Test Cases

Test Case

Comment

Input Name	Data Type	Input Value
Setup VI		
VI under Test		
<input checked="" type="checkbox"/> Over High Limit	Boolean	TRUE
<input checked="" type="checkbox"/> Under Low Limit	Boolean	TRUE
<input checked="" type="checkbox"/> Alarm reference	Refnum	[...]

Output Name	Data Type	Comparison	Expected
VI under Test			
<input checked="" type="checkbox"/> Alarm String	String	=	B
Tearardown VI			

Over High Lin
Under Low Lin
Alarm referen

Test Case Result

Test case failed.

Results Details

VI Under Test
Alarm String

Description

Value comparison failed.
 Details:
 Control Label: Alarm String
 Operation: =
 Resulting Value: OVER TEMP
 Expected Value: BOTH ALARMS

OK

OK

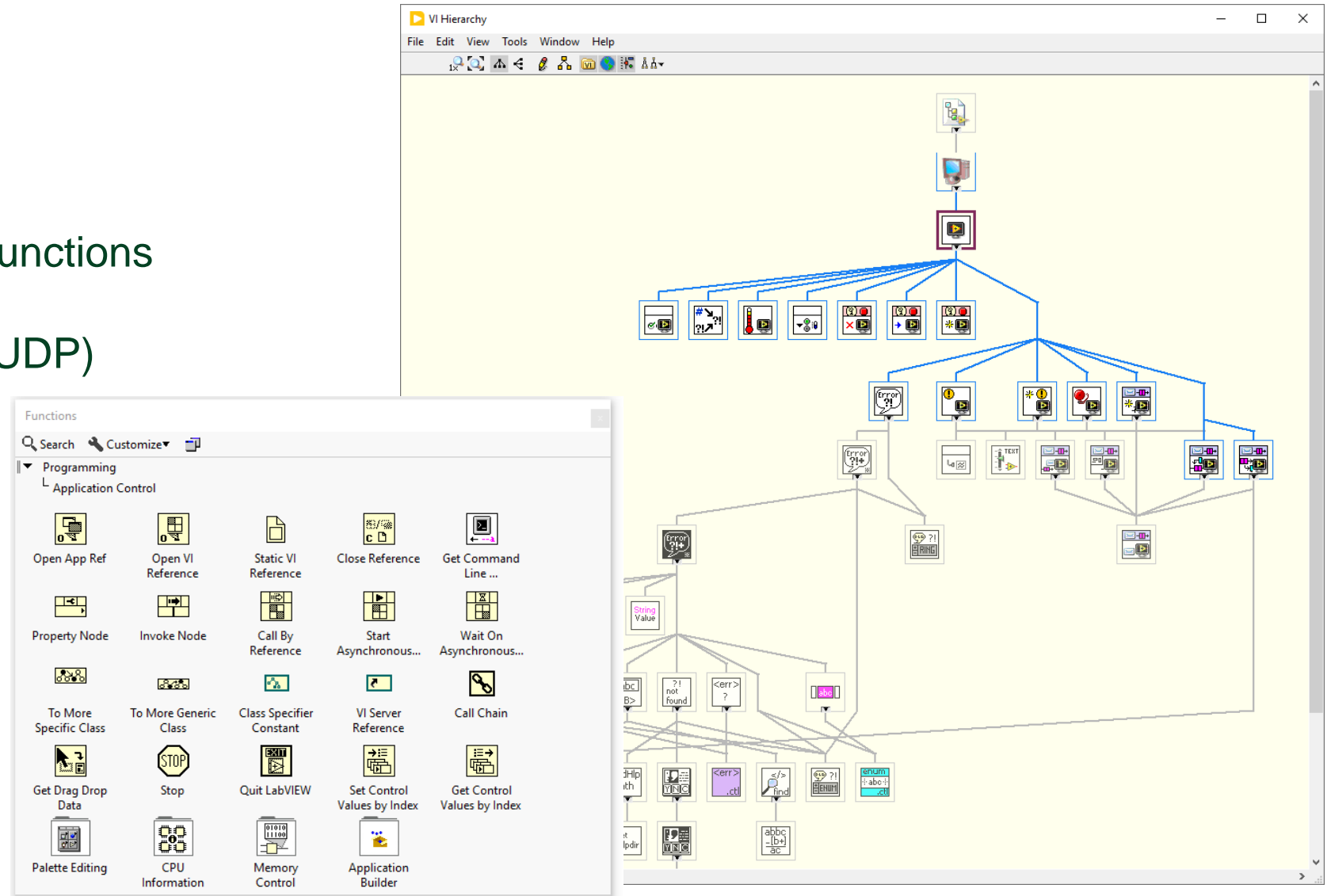
LabVIEW Static Code Analysis – Manual Search

VIEW...VI Hierarchy

See all functions being called

Review for VI calls of secure functions

- Open Ports
- Communication VIs (TCP, UDP)
- Encryption VIs
- Application Control VIs
- Command Line Interface



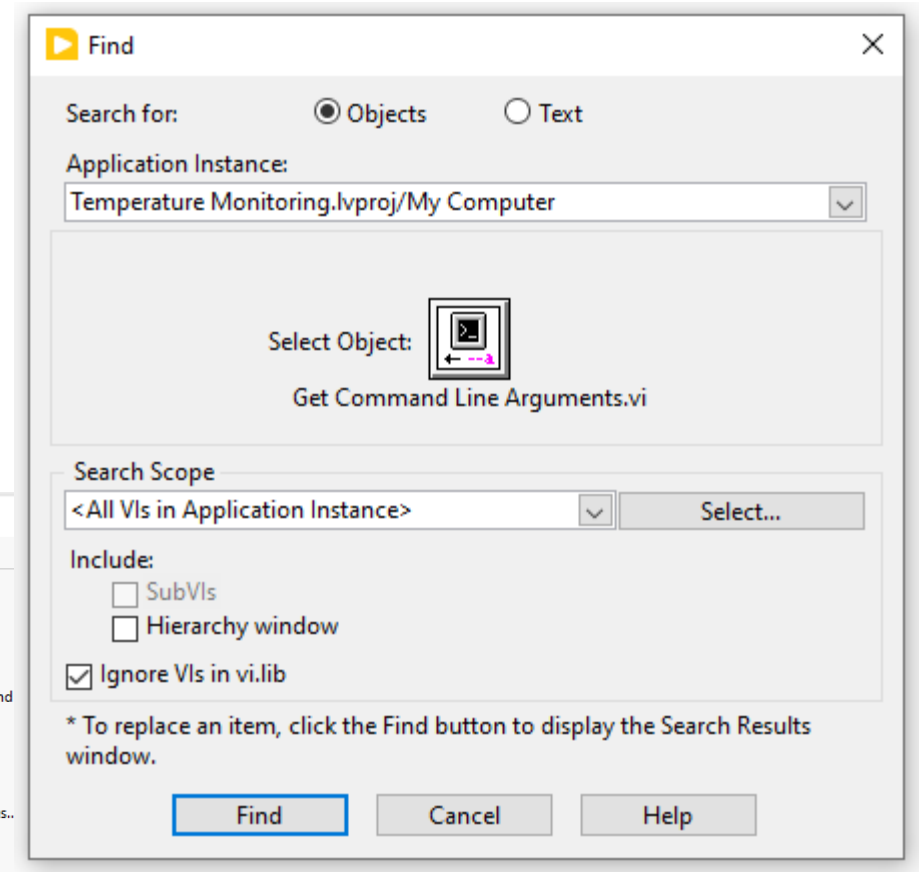
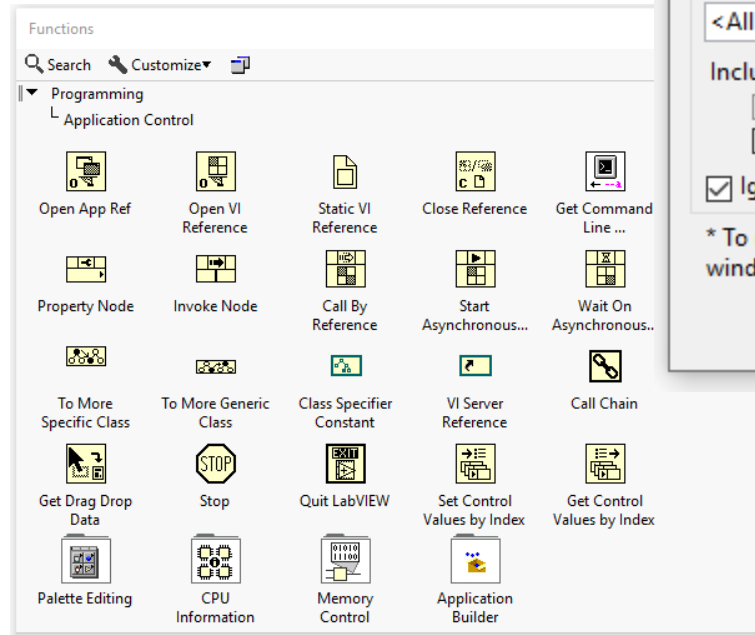
LabVIEW Static Code Analysis – LabVIEW Search

Edit...Find and Replace

Search for functions within a VI hierarchy

Search for VI calls of secure functions

- Open Ports
- Communication VIs (TCP, UDP)
- Encryption VIs
- Application Control VIs
- Command Line Interface



LabVIEW Static Code Analysis - VI Analyzer

Included with Professional; separate install

Tools...Analyze Vis

Interactive or programmatic execution

Static code analysis

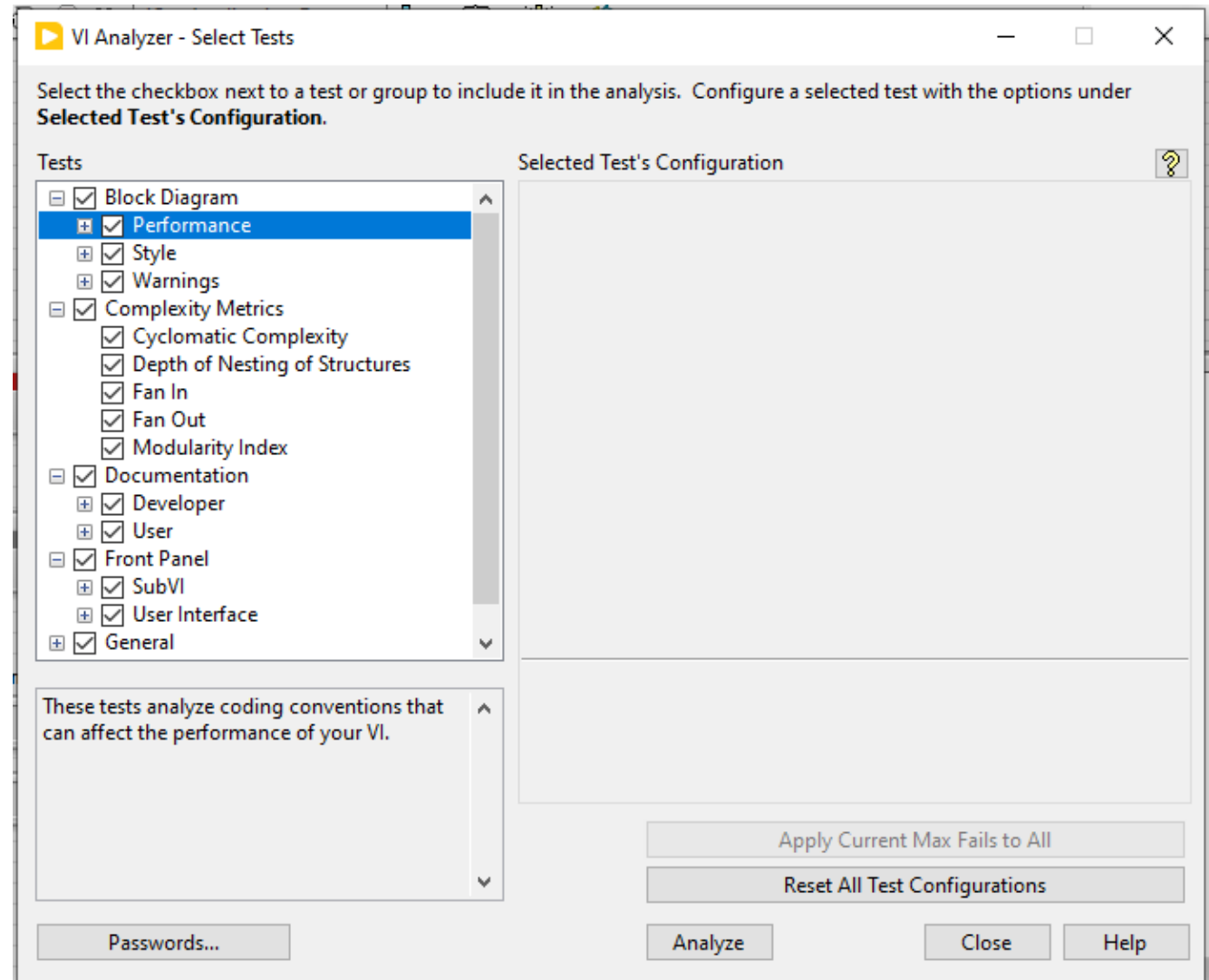
- Checks style compliance
- Identifies performance issues
- Finds bugs

Select built-in tests to run

Create custom tests to run

VI Analyzer Enthusiasts group maintains repository of custom tests

ni.com/vianalyzer





LabVIEW Static Code Analysis - VI Analyzer

VI Analyzer Results Window

Test Results Testing Errors Summary

Double-click a failure occurrence to highlight the failure object in the VI. If the VI changed since you performed the analysis, the wrong object may be highlighted.

Results List	Subitems
Simulate Temperature Acquisition.vi	7
! Hidden Objects in Structures	1
! 1: This While Loop is not entirely within the visible	
! Unused Code	1
1: This case structure has a constant wired to the	
+ Wire Bends	1
+ Wires Under Objects	2
1: This Array wire runs under an object in the "T	
2: This String wire runs under an object in the "T	
+ i Array Constant Style	1
+ i Enabled Debugging	1
+ i Error Cluster Wired	1

Show Failures Only Sort by VI ! - High Rank Test
 Sort by Test i - Low Rank Test

Description
This While Loop is not entirely within the visible bounds of its owner (While Loop).

Export... Load... Save... Done Help

Block Diagram on Temperature Monitoring.lvproj/My Computer

Tools Window Help

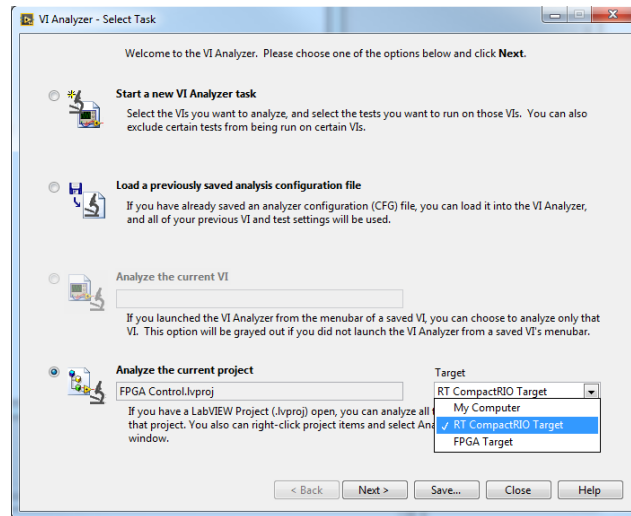
15pt Application Font Search

The block diagram shows a temperature monitoring loop. A data table on the left lists temperature values: 58984375, 58984375, 58984375, 58984375, 58984375, 58984375, 58984375, 58984375, 56640625, 51953125, 984375. The loop contains a 'While Loop' structure. A dashed blue box highlights the top portion of the loop, indicating the error 'This While Loop is not entirely within the visible bounds of its owner'. The loop body includes a 'Temperature' input, a 'Case Structure' with a constant wired to it, an 'Add' node (+1), a 'Resistor' node (R/10), and a 'Temperature' output. A '500' constant is also present in the loop.

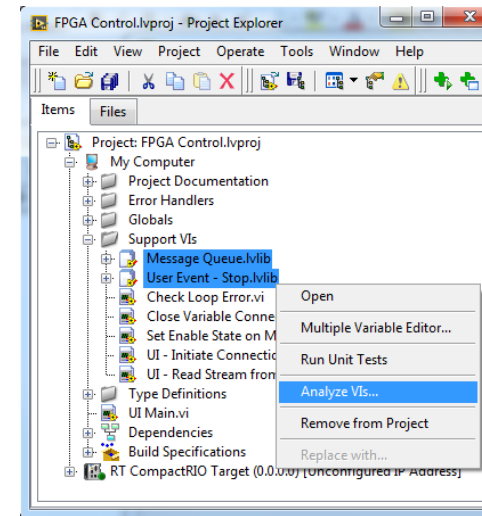


Using LabVIEW VI Analyzer for Security Tests

Run VI Analyzer from the project window to analyze all VIs contained in a project



or



Using LabVIEW VI Analyzer for Security Tests

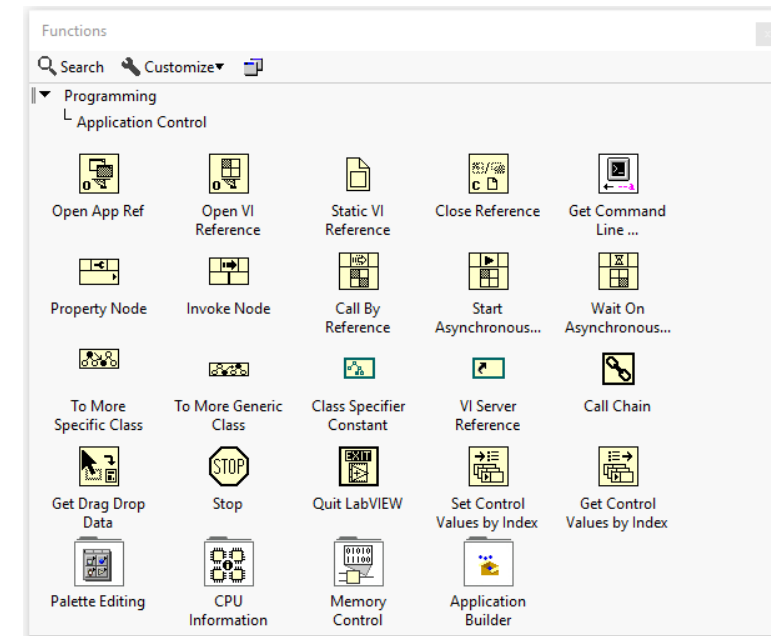
- The **VI Analyzer Test Creator** allows you to create your own tests that plug in to the VI Analyzer
- The Test Creator is a feature of the VI Analyzer Toolkit

- Configure to look for secure functions

- Open Ports
- Communication VIs (TCP, UDP)
- Encryption VIs
- Application Control VIs
- Command Line Interface

- For instructions on how to set up custom tests:

https://zone.ni.com/reference/en-XX/help/371361R-01/lvvianalyzerhelp/writing_vian_test/



Validate Code Usage

File Integrity Monitoring

Available from third-party sources

Once built into applications, LabVIEW files are managed the same as any other application

FIM tools can detect files that have been altered, updated, or compromised

A complete FIM strategy monitors:

- User-built LabVIEW applications
- Configuration files
- LabVIEW run-time engine files



FIM tools monitor properties of files, including:

- Created, modified, and accessed settings and permissions
- Security and privilege settings
- Content of the file
- Core attributes and size
- Hash values, based on file contents
- Configuration values
- Credentials



Thank you...

Steve Summers

Security Lead, NI Aerospace & Defense

steve.summers@ni.com

ni.com/security